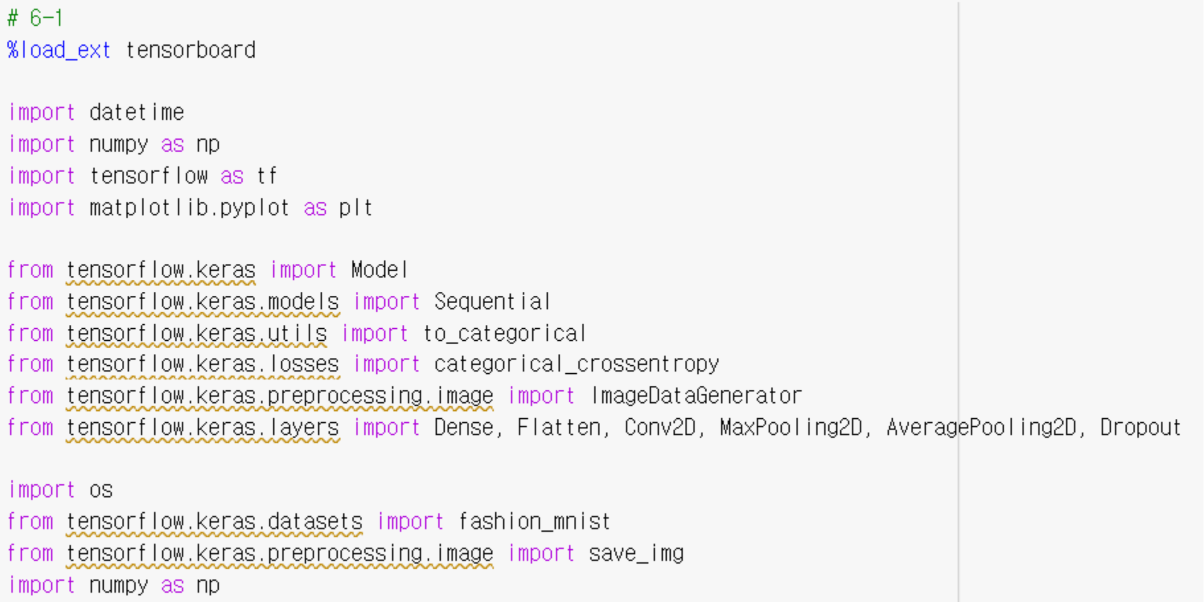
딥러닝 레포트 – 3

학번 : 2019305061

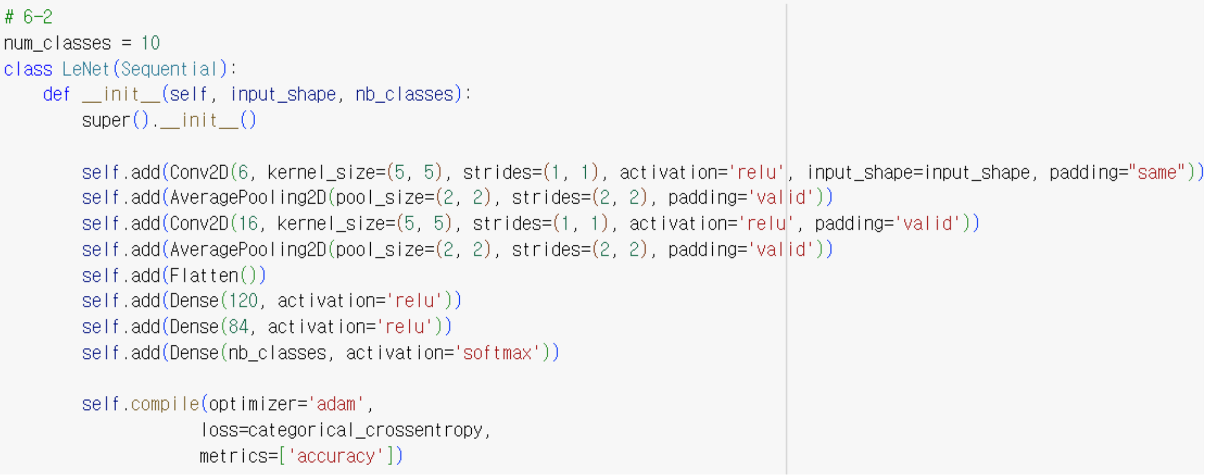
학과 : 컴퓨터공학과

이름 : 임주형

1.



```
# 6-1
%load_ext tensorboard

import datetime
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, AveragePooling2D, Dropout

import os
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.preprocessing.image import save_img
import numpy as np
```



```
# 6-2
num_classes = 10
class LeNet(Sequential):
    def __init__(self, input_shape, nb_classes):
        super().__init__()

        self.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='relu', input_shape=input_shape, padding="same"))
        self.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
        self.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='valid'))
        self.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
        self.add(Flatten())
        self.add(Dense(120, activation='relu'))
        self.add(Dense(84, activation='relu'))
        self.add(Dense(nb_classes, activation='softmax'))

        self.compile(optimizer='adam',
                     loss=categorical_crossentropy,
                     metrics=['accuracy'])
```

```
# 6-3
model = LeNet((28, 28, 1), num_classes)
model.summary()
```

```
Model: "le_net"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         156

 average_pooling2d (Average  (None, 14, 14, 6)         0
 Pooling2D)

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_1 (Avera  (None, 5, 5, 16)          0
 gePooling2D)

 flatten (Flatten)           (None, 400)               0

 dense (Dense)               (None, 120)               48120

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 10)                850

=================================================================
Total params: 61706 (241.04 KB)
Trainable params: 61706 (241.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

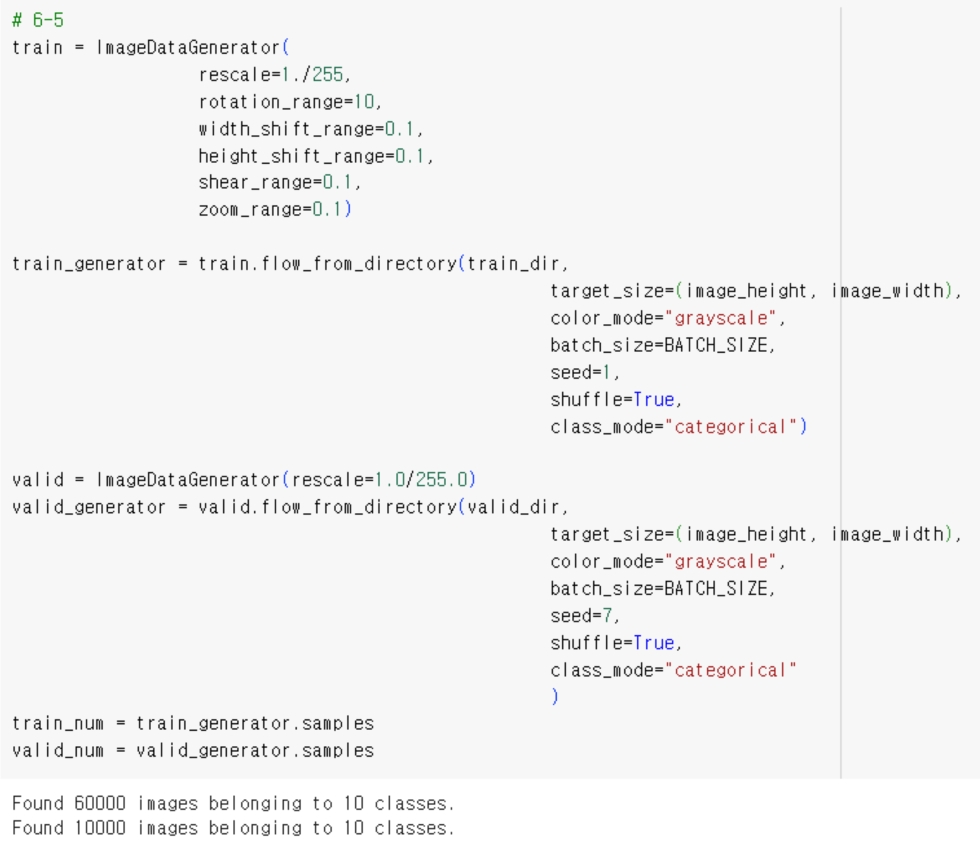
```
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 1s 0us/step
```

```
# 6-4
EPOCHS = 5
BATCH_SIZE = 32
image_height = 28
image_width = 28

# 저장 디렉토리 생성
train_dir = '/content/train_dir'
os.makedirs(train_dir, exist_ok=True)
valid_dir = '/content/valid_dir'
os.makedirs(valid_dir, exist_ok=True)
```

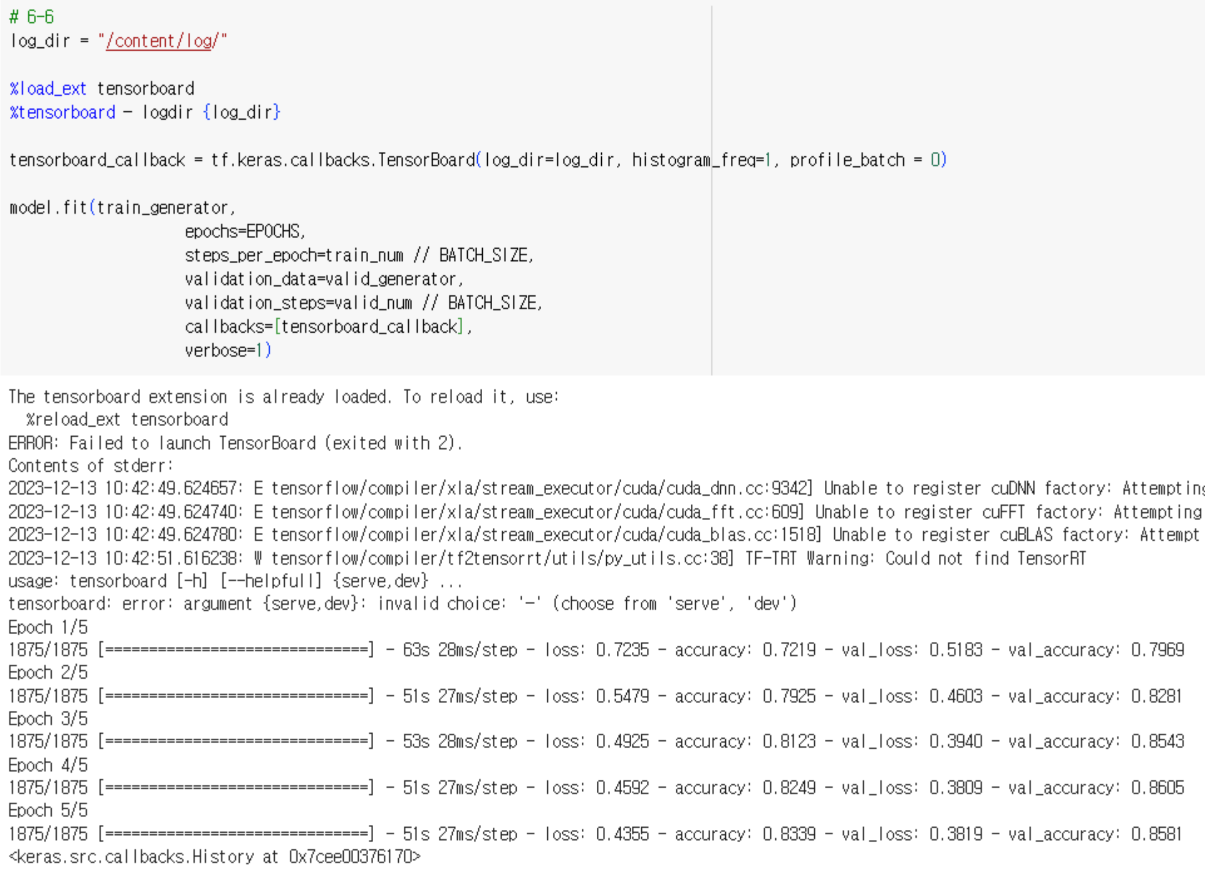
```
# x_train 을 label 디렉토리 생성하여 label별로 저장
for i in range(len(y_train)):
 if os.path.exists('/content/train_dir/' + str(y_train[i])) ==False :
  class_dir = os.path.join(train_dir, str(y_train[i]))
  os.makedirs(class_dir, exist_ok=True)
 image_path = os.path.join('/content/train_dir/' + str(y_train[i]), f'{i}.png')
 save_img(image_path, np.expand_dims(x_train[i], axis=-1))
```

```
#x_test 를 label 디렉토리 생성하여 label별로 저장
for i in range(len(y_test)):
 if os.path.exists('/content/valid_dir/' + str(y_test[i])) ==False :
  class_dir = os.path.join(valid_dir, str(y_test[i]))
  os.makedirs(class_dir, exist_ok=True)
 image_path = os.path.join('/content/valid_dir/' + str(y_test[i]), f'{i}.png')
 save_img(image_path, np.expand_dims(x_test[i], axis=-1))
```



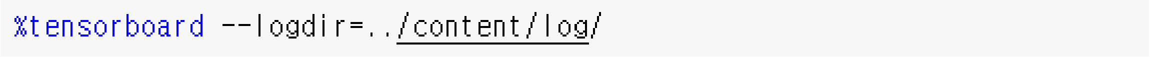
```
# 6-5
train = ImageDataGenerator(
                  rescale=1./255,
                  rotation_range=10,
                  width_shift_range=0.1,
                  height_shift_range=0.1,
                  shear_range=0.1,
                  zoom_range=0.1)

train_generator = train.flow_from_directory(train_dir,
                                                       target_size=(image_height, image_width),
                                                       color_mode="grayscale",
                                                       batch_size=BATCH_SIZE,
                                                       seed=1,
                                                       shuffle=True,
                                                       class_mode="categorical")

valid = ImageDataGenerator(rescale=1.0/255.0)
valid_generator = valid.flow_from_directory(valid_dir,
                                                       target_size=(image_height, image_width),
                                                       color_mode="grayscale",
                                                       batch_size=BATCH_SIZE,
                                                       seed=7,
                                                       shuffle=True,
                                                       class_mode="categorical"
                                                       )
train_num = train_generator.samples
valid_num = valid_generator.samples
```

```
Found 60000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
```

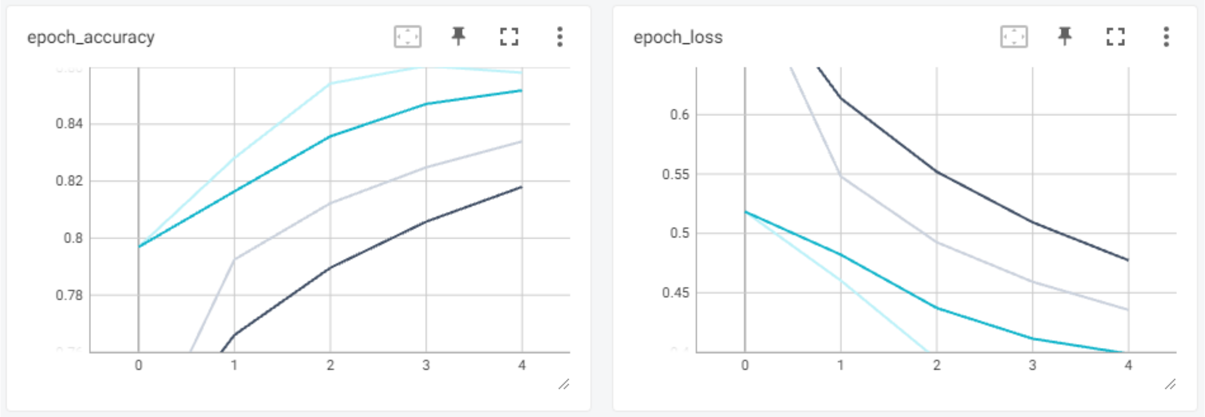


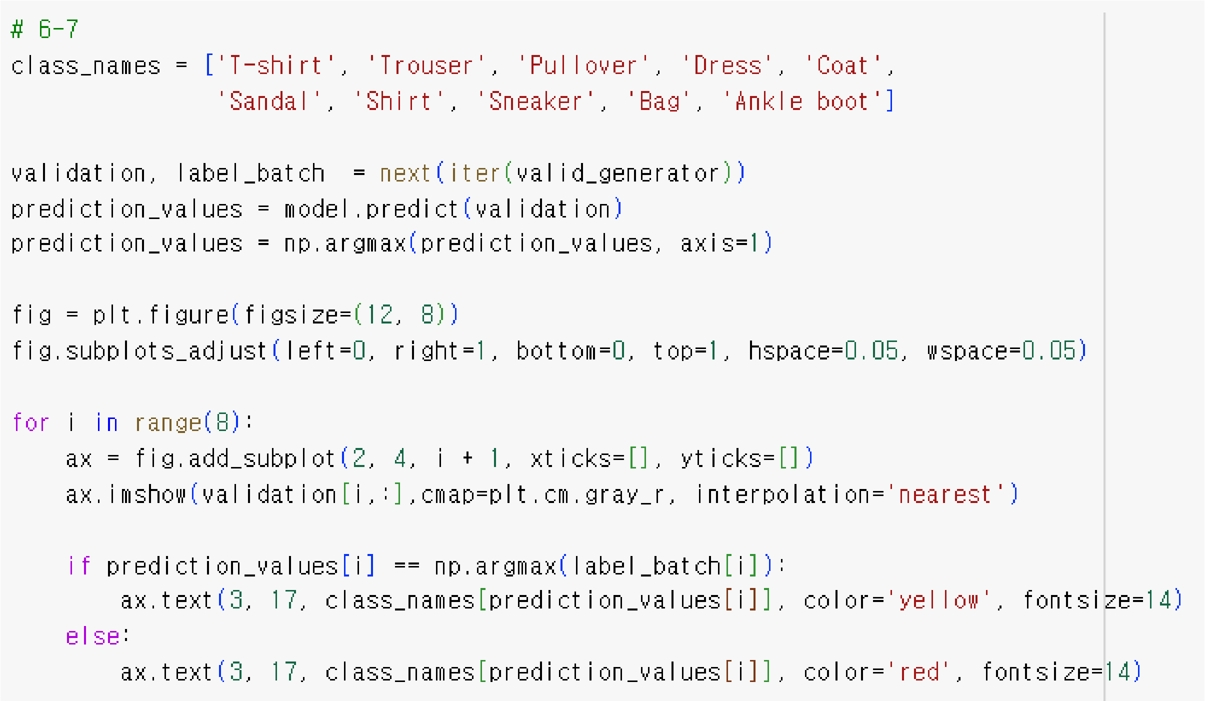
```
# 6-6
log_dir = "/content/log/"

%load_ext tensorboard
%tensorboard - logdir {log_dir}

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1, profile_batch = 0)

model.fit(train_generator,
                    epochs=EPOCHS,
                    steps_per_epoch=train_num // BATCH_SIZE,
                    validation_data=valid_generator,
                    validation_steps=valid_num // BATCH_SIZE,
                    callbacks=[tensorboard_callback],
                    verbose=1)
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
ERROR: Failed to launch TensorBoard (exited with 2).
Contents of stderr:
2023-12-13 10:42:49.624657: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attemptin
2023-12-13 10:42:49.624740: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting
2023-12-13 10:42:49.624780: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempt
2023-12-13 10:42:51.616238: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
usage: tensorboard [-h] [--helpfull] {serve,dev} ...
tensorboard: error: argument {serve,dev}: invalid choice: '-' (choose from 'serve', 'dev')
Epoch 1/5
1875/1875 [==============================] - 63s 28ms/step - loss: 0.7235 - accuracy: 0.7219 - val_loss: 0.5183 - val_accuracy: 0.7969
Epoch 2/5
1875/1875 [==============================] - 51s 27ms/step - loss: 0.5479 - accuracy: 0.7925 - val_loss: 0.4603 - val_accuracy: 0.8281
Epoch 3/5
1875/1875 [==============================] - 53s 28ms/step - loss: 0.4925 - accuracy: 0.8123 - val_loss: 0.3940 - val_accuracy: 0.8543
Epoch 4/5
1875/1875 [==============================] - 51s 27ms/step - loss: 0.4592 - accuracy: 0.8249 - val_loss: 0.3809 - val_accuracy: 0.8605
Epoch 5/5
1875/1875 [==============================] - 51s 27ms/step - loss: 0.4355 - accuracy: 0.8339 - val_loss: 0.3819 - val_accuracy: 0.8581
<keras.src.callbacks.History at 0x7cee00376170>
```



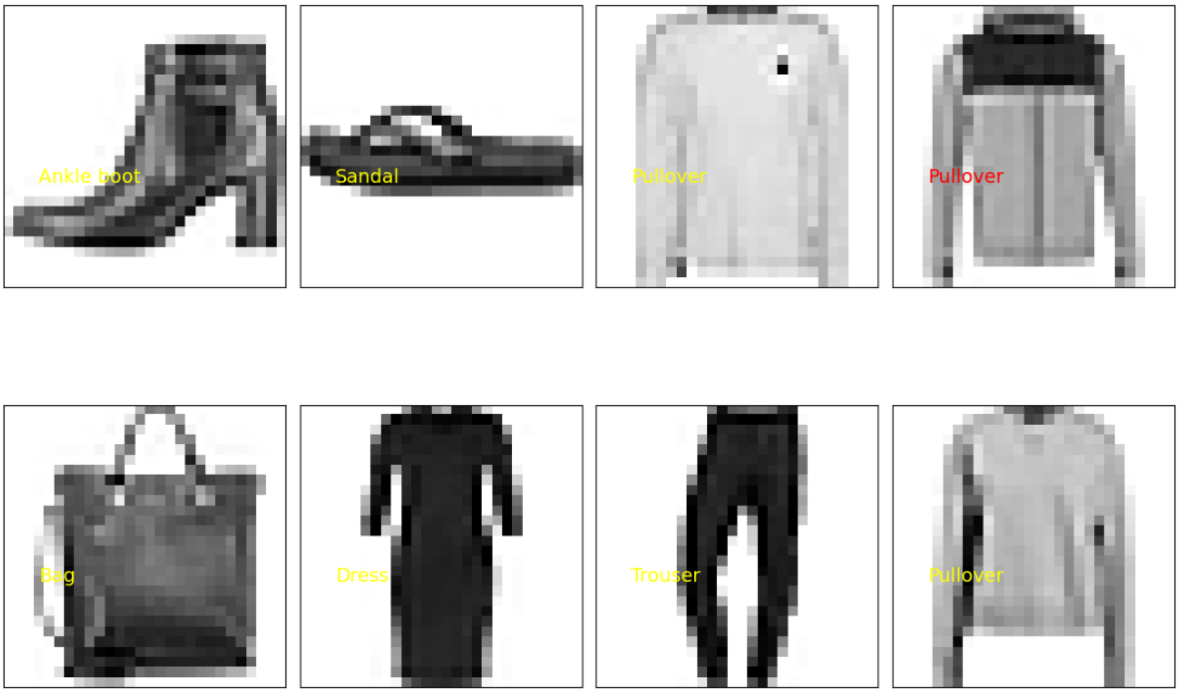
```
%tensorboard --logdir=../content/log/
```



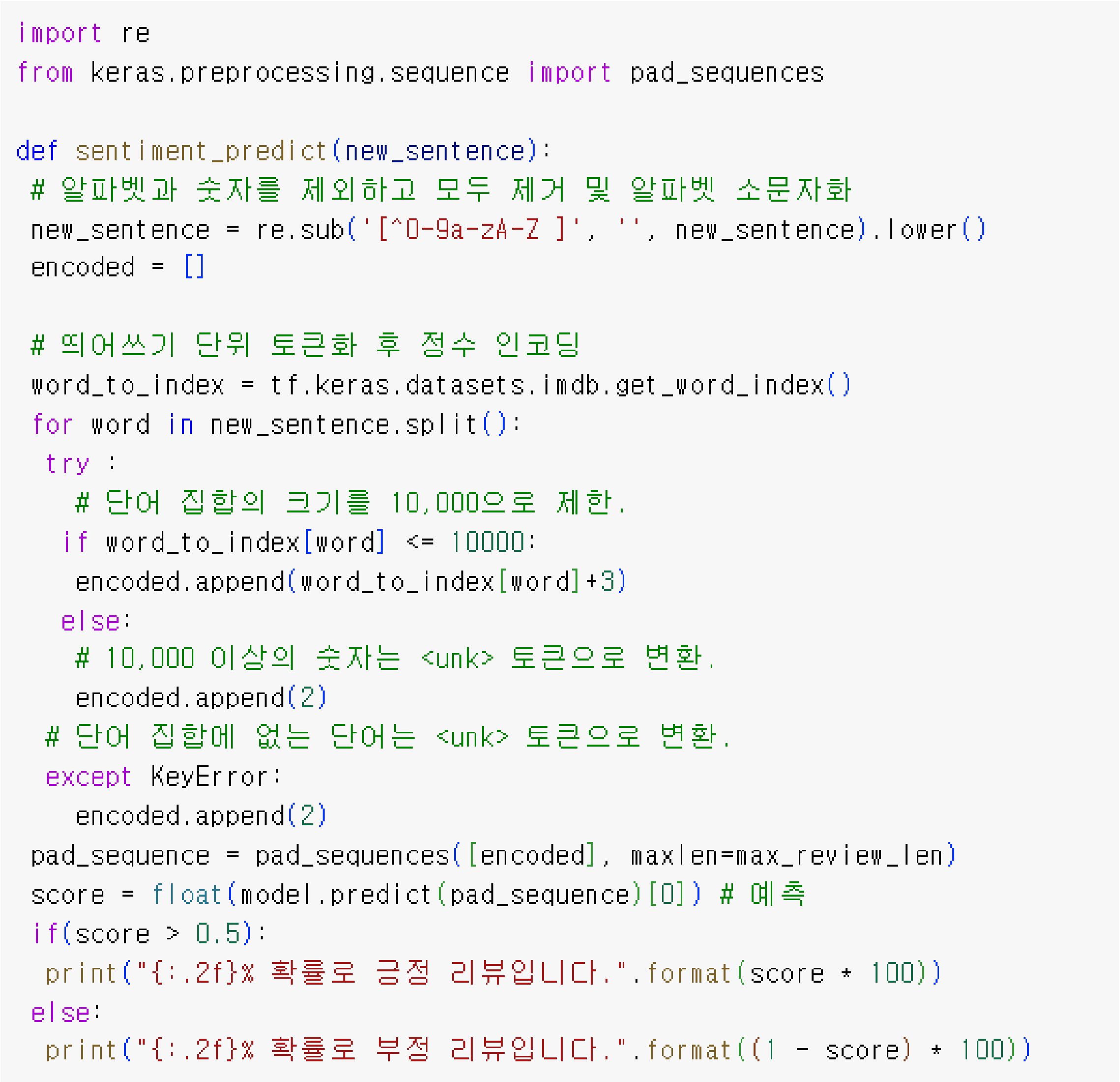


```
# 6-7
class_names = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

validation, label_batch  = next(iter(valid_generator))
prediction_values = model.predict(validation)
prediction_values = np.argmax(prediction_values, axis=1)

fig = plt.figure(figsize=(12, 8))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(8):
    ax = fig.add_subplot(2, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(validation[i,:],cmap=plt.cm.gray_r, interpolation='nearest')

    if prediction_values[i] == np.argmax(label_batch[i]):
        ax.text(3, 17, class_names[prediction_values[i]], color='yellow', fontsize=14)
    else:
        ax.text(3, 17, class_names[prediction_values[i]], color='red', fontsize=14)
```

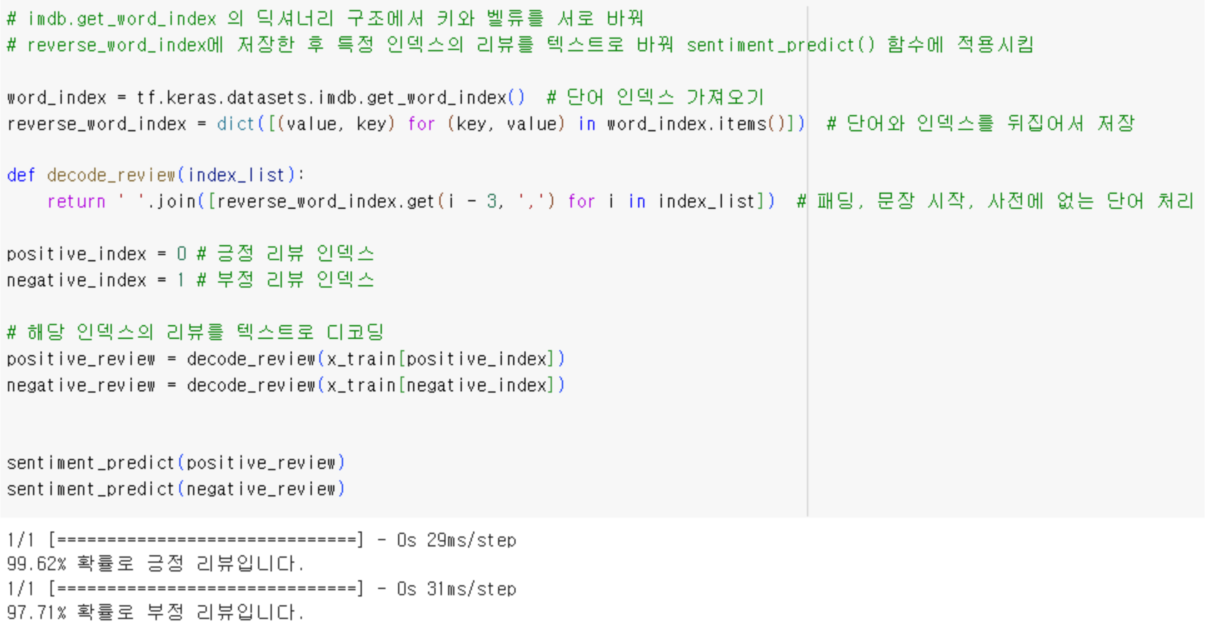
```
1/1 [==============================] - 0s 25ms/step
```



2.



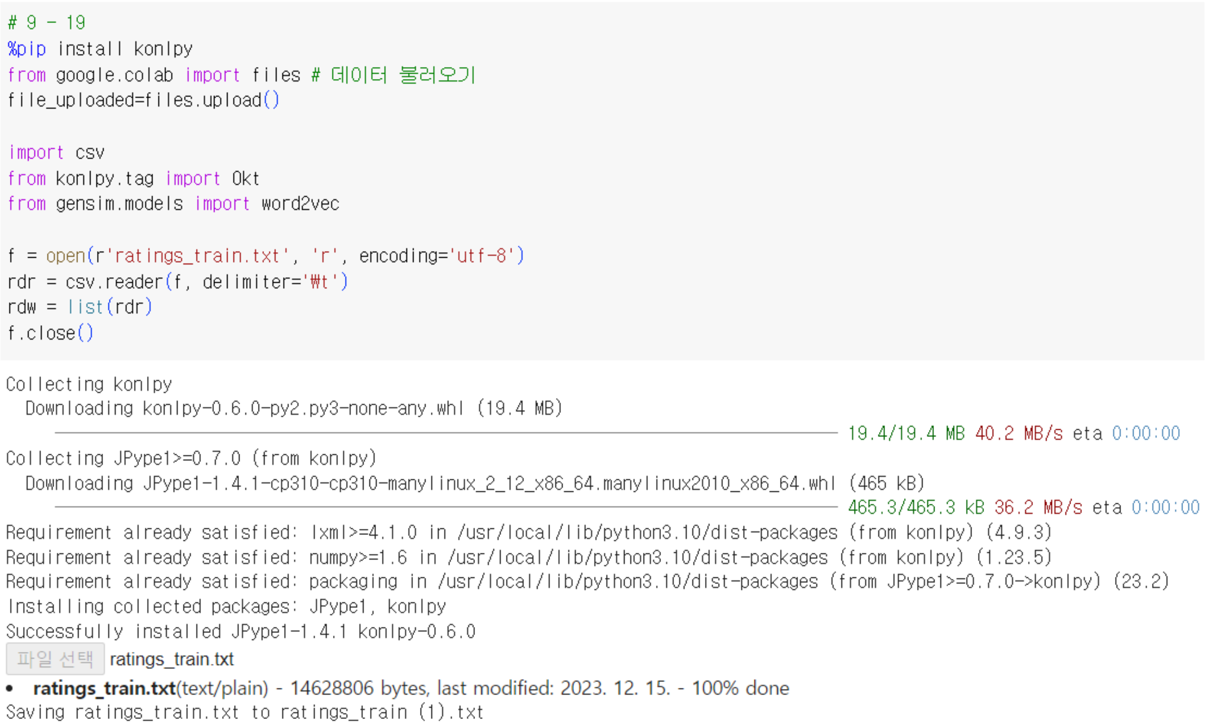
```
import re
from keras.preprocessing.sequence import pad_sequences

def sentiment_predict(new_sentence):
 # 알파벳과 숫자를 제외하고 모두 제거 및 알파벳 소문자화
 new_sentence = re.sub('[^0-9a-zA-Z ]', '', new_sentence).lower()
 encoded = []

 # 띄어쓰기 단위 토큰화 후 정수 인코딩
 word_to_index = tf.keras.datasets.imdb.get_word_index()
 for word in new_sentence.split():
  try :
    # 단어 집합의 크기를 10,000으로 제한.
   if word_to_index[word] <= 10000:
    encoded.append(word_to_index[word]+3)
   else:
    # 10,000 이상의 숫자는 <unk> 토큰으로 변환.
    encoded.append(2)
  # 단어 집합에 없는 단어는 <unk> 토큰으로 변환.
  except KeyError:
    encoded.append(2)
 pad_sequence = pad_sequences([encoded], maxlen=max_review_len)
 score = float(model.predict(pad_sequence)[0]) # 예측
 if(score > 0.5):
  print("{:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
 else:
  print("{:.2f}% 확률로 부정 리뷰입니다.".format((1 - score) * 100))
```



```
# imdb.get_word_index 의 딕셔너리 구조에서 키와 벨류를 서로 바꿔
# reverse_word_index에 저장한 후 특정 인덱스의 리뷰를 텍스트로 바꿔 sentiment_predict() 함수에 적용시킴

word_index = tf.keras.datasets.imdb.get_word_index()  # 단어 인덱스 가져오기
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])  # 단어와 인덱스를 뒤집어서 저장

def decode_review(index_list):
    return ' '.join([reverse_word_index.get(i - 3, ',') for i in index_list])  # 패딩, 문장 시작, 사전에 없는 단어 처리

positive_index = 0 # 긍정 리뷰 인덱스
negative_index = 1 # 부정 리뷰 인덱스

# 해당 인덱스의 리뷰를 텍스트로 디코딩
positive_review = decode_review(x_train[positive_index])
negative_review = decode_review(x_train[negative_index])


sentiment_predict(positive_review)
sentiment_predict(negative_review)
```

```
1/1 [==============================] - 0s 29ms/step
99.62% 확률로 긍정 리뷰입니다.
1/1 [==============================] - 0s 31ms/step
97.71% 확률로 부정 리뷰입니다.
```

3.



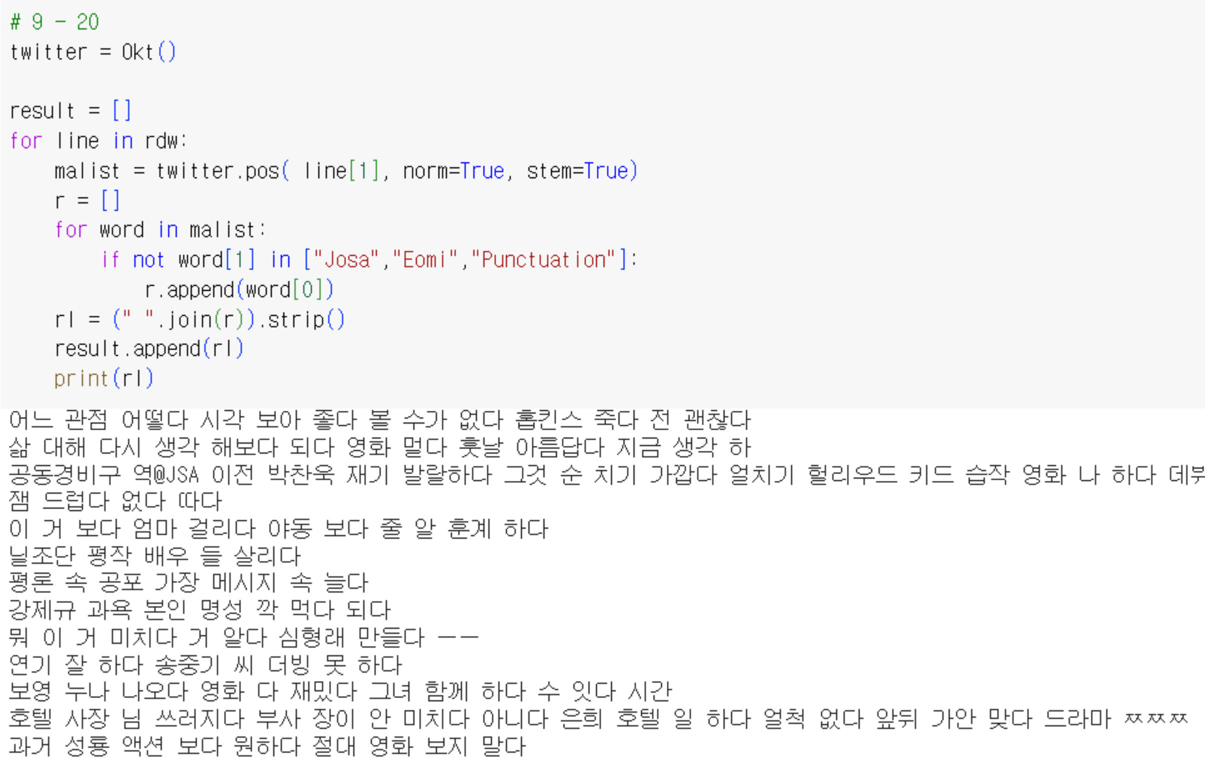
```
# 9 - 19
%pip install konlpy
from google.colab import files # 데이터 불러오기
file_uploaded=files.upload()

import csv
from konlpy.tag import Okt
from gensim.models import word2vec

f = open(r'ratings_train.txt', 'r', encoding='utf-8')
rdr = csv.reader(f, delimiter='\t')
rdw = list(rdr)
f.close()
```

```
Collecting konlpy
  Downloading konlpy-0.6.0-py2.py3-none-any.whl (19.4 MB)
     ────────────────────────────────────────── 19.4/19.4 MB 40.2 MB/s eta 0:00:00
Collecting JPype1>=0.7.0 (from konlpy)
  Downloading JPype1-1.4.1-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (465 kB)
     ────────────────────────────────────────── 465.3/465.3 kB 36.2 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (4.9.3)
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.23.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from JPype1>=0.7.0->konlpy) (23.2)
Installing collected packages: JPype1, konlpy
Successfully installed JPype1-1.4.1 konlpy-0.6.0
```

ratings_train.txt

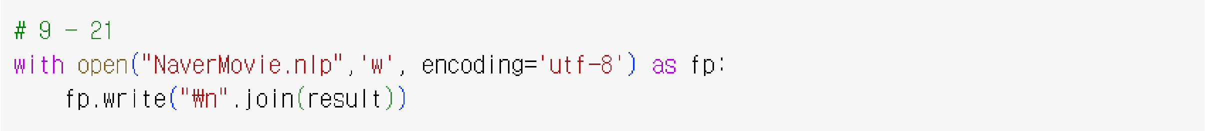
- **ratings_train.txt**(text/plain) - 14628806 bytes, last modified: 2023. 12. 15. - 100% done

```
Saving ratings_train.txt to ratings_train (1).txt
```

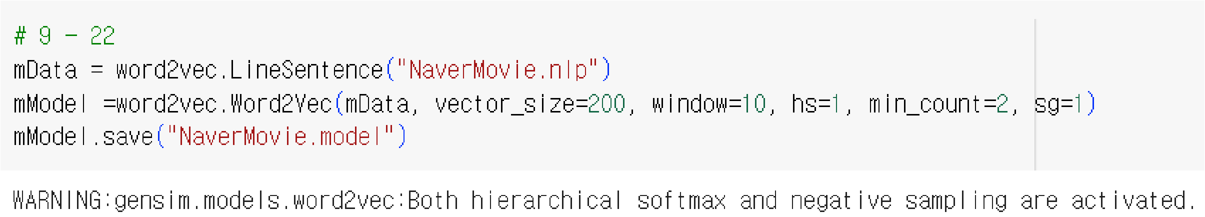


```
# 9 - 20
twitter = Okt()

result = []
for line in rdw:
    malist = twitter.pos( line[1], norm=True, stem=True)
    r = []
    for word in malist:
        if not word[1] in ["Josa","Eomi","Punctuation"]:
            r.append(word[0])
    rl = (" ".join(r)).strip()
    result.append(rl)
    print(rl)
```

```
어느 관점 어떨다 시각 보아 좋다 볼 수가 없다 홉킨스 죽다 전 괜찮다
삶 대해 다시 생각 해보다 되다 영화 멀다 훗날 아름답다 지금 생각 하
공동경비구 역@JSA 이전 박찬욱 재기 발랄하다 그것 순 치기 가깝다 얼치기 헐리우드 키드 습작 영화 나 하다 데뷔
잼 드럽다 없다 따다
이 거 보다 엄마 걸리다 야동 보다 줄 알 훈계 하다
닐조단 평작 배우 들 살리다
평론 속 공포 가장 메시지 속 늘다
강제규 과욕 본인 명성 깍 먹다 되다
뭐 이 거 미치다 거 알다 심형래 만들다 --
연기 잘 하다 송중기 씨 더빙 못 하다
보영 누나 나오다 영화 다 재밌다 그녀 함께 하다 수 있다 시간
호텔 사장 님 쓰러지다 부사 장이 안 미치다 아니다 은희 호텔 일 하다 얼척 없다 앞뒤 가안 맞다 드라마 ㅉㅉㅉ
과거 성룡 액션 보다 원하다 절대 영화 보지 말다
```

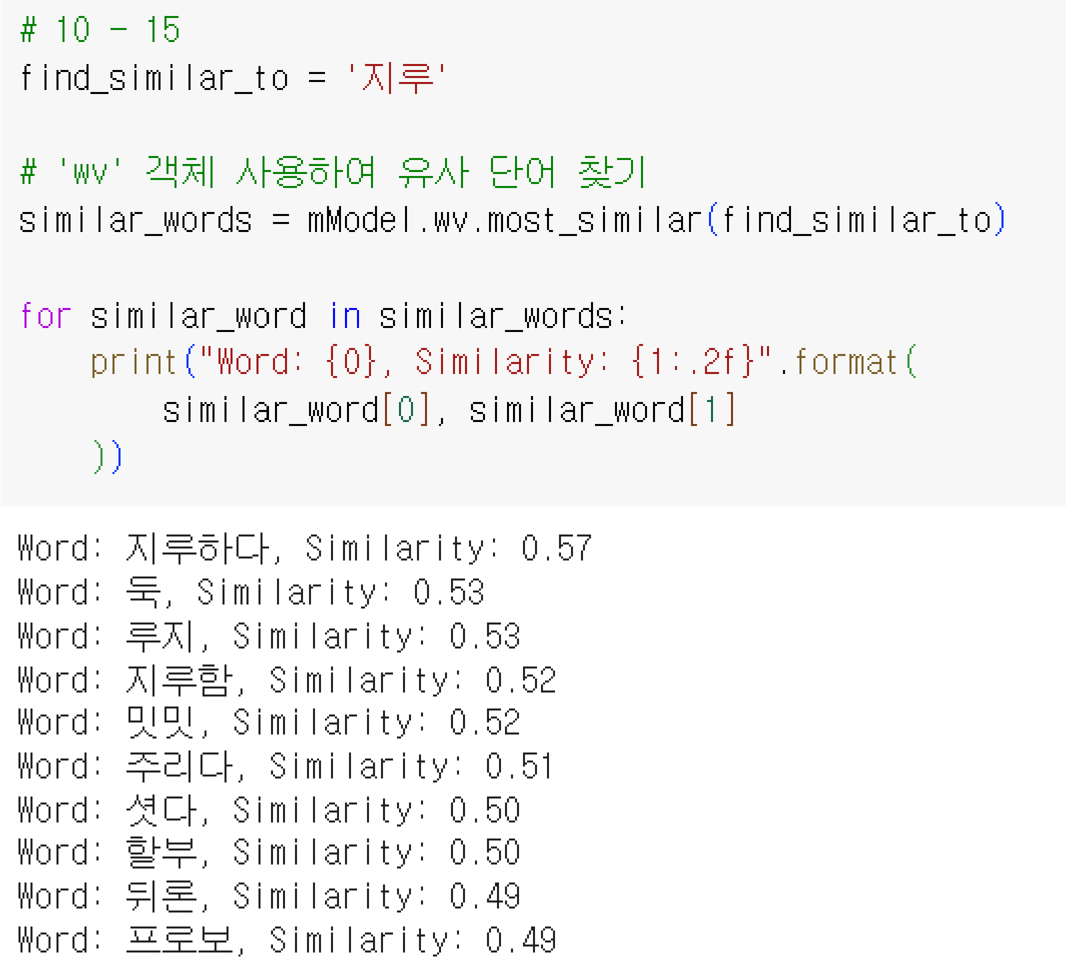


```
# 9 - 21
with open("NaverMovie.nlp",'w', encoding='utf-8') as fp:
    fp.write("₩n".join(result))
```



```
# 9 - 22
mData = word2vec.LineSentence("NaverMovie.nlp")
mModel =word2vec.Word2Vec(mData, vector_size=200, window=10, hs=1, min_count=2, sg=1)
mModel.save("NaverMovie.model")
```

```
WARNING:gensim.models.word2vec:Both hierarchical softmax and negative sampling are activated.
```



```
# 10 - 15
find_similar_to = '지루'

# 'wv' 객체 사용하여 유사 단어 찾기
similar_words = mModel.wv.most_similar(find_similar_to)

for similar_word in similar_words:
    print("Word: {0}, Similarity: {1:.2f}".format(
        similar_word[0], similar_word[1]
    ))
```

```
Word: 지루하다, Similarity: 0.57
Word: 둑, Similarity: 0.53
Word: 루지, Similarity: 0.53
Word: 지루함, Similarity: 0.52
Word: 밋밋, Similarity: 0.52
Word: 주리다, Similarity: 0.51
Word: 셧다, Similarity: 0.50
Word: 할부, Similarity: 0.50
Word: 뒤론, Similarity: 0.49
Word: 프로보, Similarity: 0.49
```

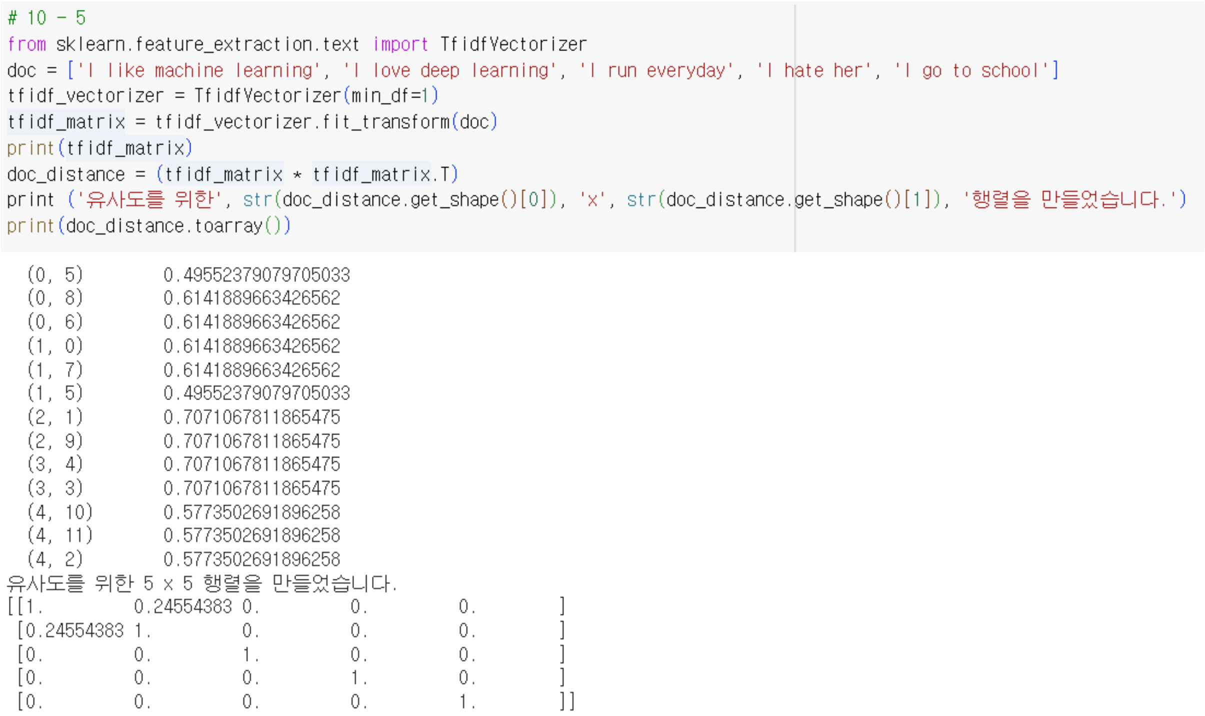
4.

```
# 10 - 1
from google.colab import files
file_uploaded = files.upload()

import pandas as pd
class2 = pd.read_csv("class2.csv")

from sklearn import preprocessing

# Label Encoding
label_encoder = preprocessing.LabelEncoder()
class2['tissue_encoded'] = label_encoder.fit_transform(class2['tissue'])
print("Label Encoded:")
print(class2['tissue_encoded'])

# One-Hot Encoding
onehot_encoder = preprocessing.OneHotEncoder(sparse=False)
encoded_onehot = onehot_encoder.fit_transform(class2[['tissue_encoded']])
print("₩nOne-Hot Encoded:")
print(encoded_onehot)
```

파일 선택 class2.csv

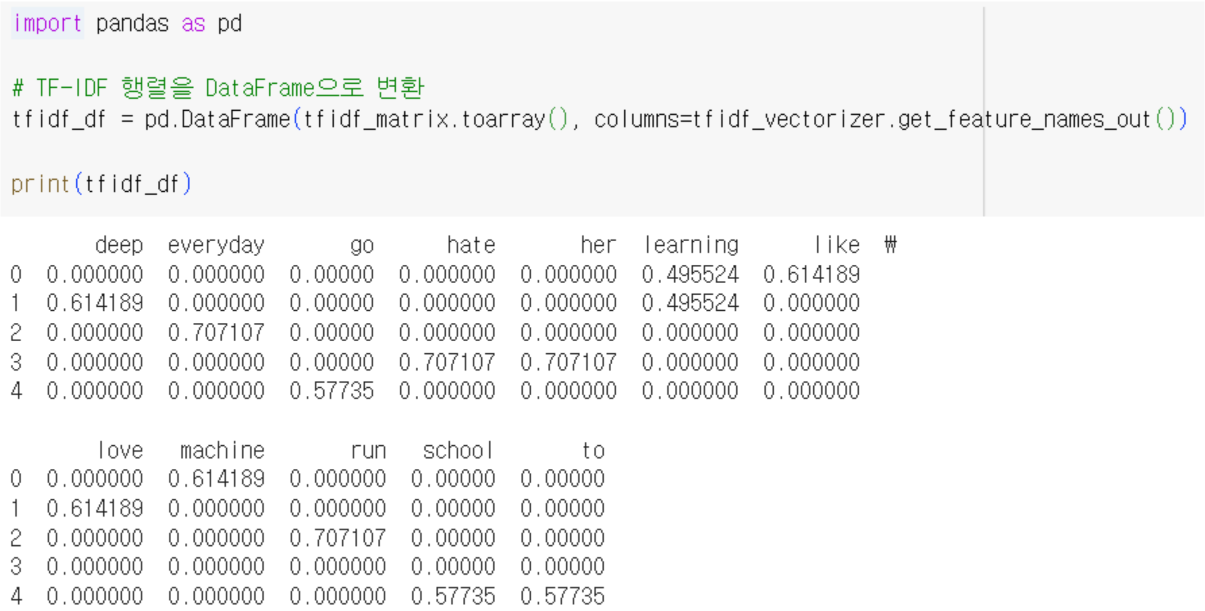
- **class2.csv**(text/csv) - 210 bytes, last modified: 2023. 12. 15. - 100% done

```
Saving class2.csv to class2.csv
Label Encoded:
0    1
1    0
2    0
3    1
4    2
5    1
Name: tissue_encoded, dtype: int64

One-Hot Encoded:
[[0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]]
```



```
# 10 - 5
from sklearn.feature_extraction.text import TfidfVectorizer
doc = ['I like machine learning', 'I love deep learning', 'I run everyday', 'I hate her', 'I go to school']
tfidf_vectorizer = TfidfVectorizer(min_df=1)
tfidf_matrix = tfidf_vectorizer.fit_transform(doc)
print(tfidf_matrix)
doc_distance = (tfidf_matrix * tfidf_matrix.T)
print ('유사도를 위한', str(doc_distance.get_shape()[0]), 'x', str(doc_distance.get_shape()[1]), '행렬을 만들었습니다.')
print(doc_distance.toarray())
```

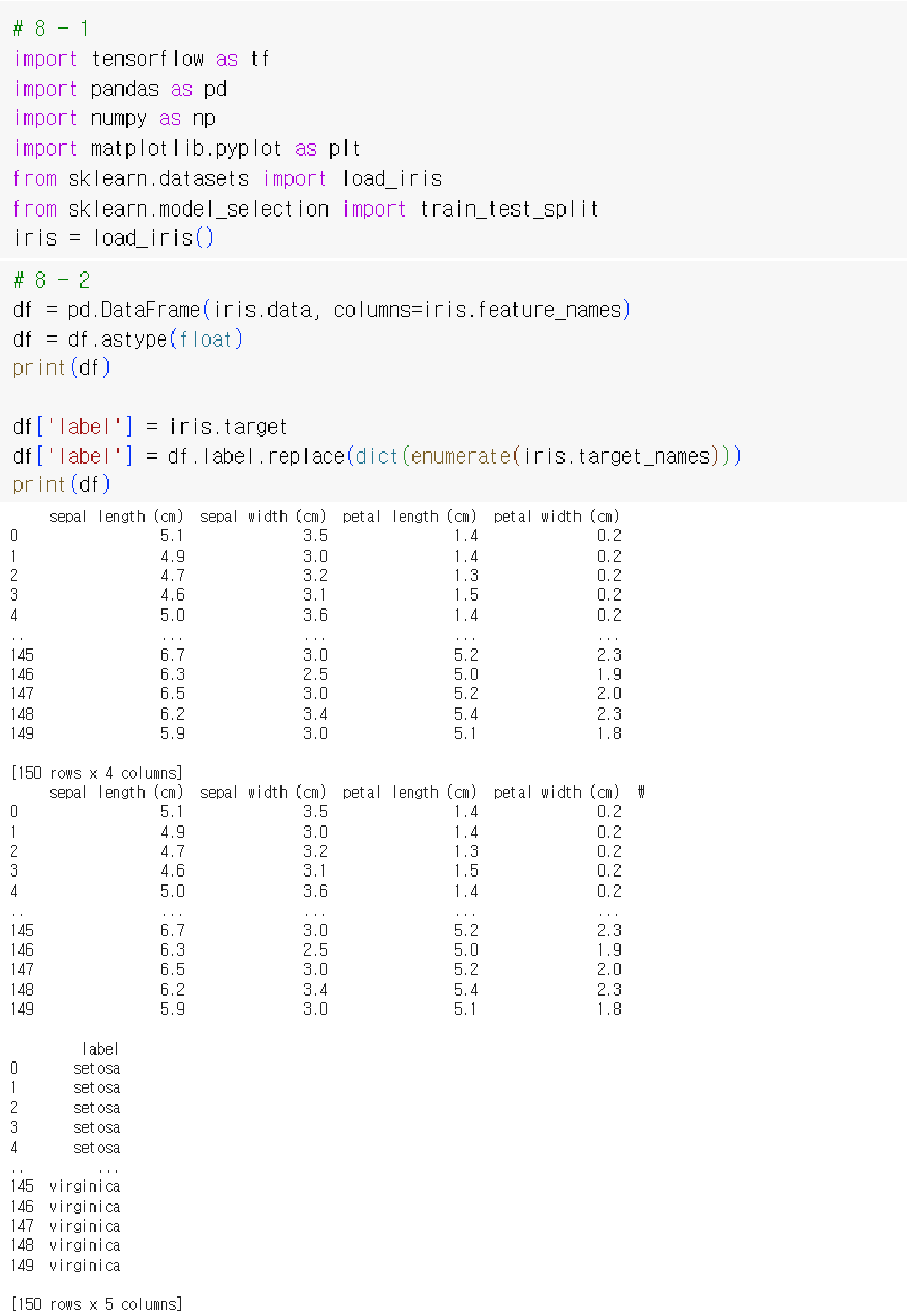
```
  (0, 5)	0.49552379079705033
  (0, 8)	0.6141889663426562
  (0, 6)	0.6141889663426562
  (1, 0)	0.6141889663426562
  (1, 7)	0.6141889663426562
  (1, 5)	0.49552379079705033
  (2, 1)	0.7071067811865475
  (2, 9)	0.7071067811865475
  (3, 4)	0.7071067811865475
  (3, 3)	0.7071067811865475
  (4, 10)	0.5773502691896258
  (4, 11)	0.5773502691896258
  (4, 2)	0.5773502691896258
유사도를 위한 5 x 5 행렬을 만들었습니다.
[[1.         0.24554383 0.         0.         0.        ]
 [0.24554383 1.         0.         0.         0.        ]
 [0.         0.         1.         0.         0.        ]
 [0.         0.         0.         1.         0.        ]
 [0.         0.         0.         0.         1.        ]]
```



```
import pandas as pd

# TF-IDF 행렬을 DataFrame으로 변환
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

print(tfidf_df)
```

```
       deep  everyday       go      hate       her  learning      like  ₩
0  0.000000  0.000000  0.00000  0.000000  0.000000  0.495524  0.614189
1  0.614189  0.000000  0.00000  0.000000  0.000000  0.495524  0.000000
2  0.000000  0.707107  0.00000  0.000000  0.000000  0.000000  0.000000
3  0.000000  0.000000  0.00000  0.707107  0.707107  0.000000  0.000000
4  0.000000  0.000000  0.57735  0.000000  0.000000  0.000000  0.000000

       love   machine       run   school       to
0  0.000000  0.614189  0.000000  0.00000  0.00000
1  0.614189  0.000000  0.000000  0.00000  0.00000
2  0.000000  0.000000  0.707107  0.00000  0.00000
3  0.000000  0.000000  0.000000  0.00000  0.00000
4  0.000000  0.000000  0.000000  0.57735  0.57735
```

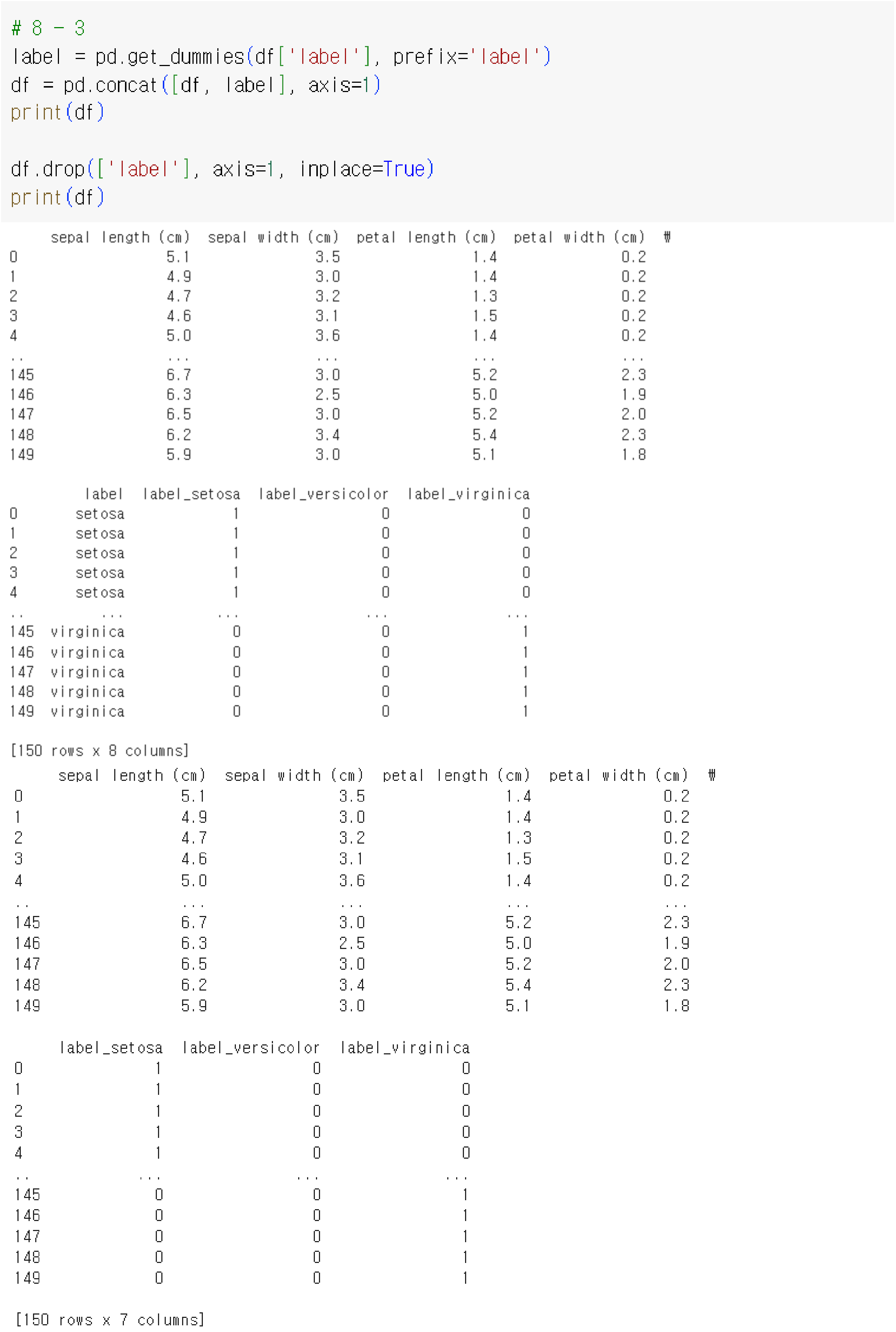
5.



```
# 8 - 1
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
```

```
# 8 - 2
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df = df.astype(float)
print(df)

df['label'] = iris.target
df['label'] = df.label.replace(dict(enumerate(iris.target_names)))
print(df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

         label
0       setosa
1       setosa
2       setosa
3       setosa
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 5 columns]
```



```python
# 8 - 3
label = pd.get_dummies(df['label'], prefix='label')
df = pd.concat([df, label], axis=1)
print(df)

df.drop(['label'], axis=1, inplace=True)
print(df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2   
1                  4.9               3.0                1.4               0.2   
2                  4.7               3.2                1.3               0.2   
3                  4.6               3.1                1.5               0.2   
4                  5.0               3.6                1.4               0.2   
..                 ...               ...                ...               ...   
145                6.7               3.0                5.2               2.3   
146                6.3               2.5                5.0               1.9   
147                6.5               3.0                5.2               2.0   
148                6.2               3.4                5.4               2.3   
149                5.9               3.0                5.1               1.8   

         label  label_setosa  label_versicolor  label_virginica  
0       setosa             1                 0                0  
1       setosa             1                 0                0  
2       setosa             1                 0                0  
3       setosa             1                 0                0  
4       setosa             1                 0                0  
..         ...           ...               ...              ...  
145  virginica             0                 0                1  
146  virginica             0                 0                1  
147  virginica             0                 0                1  
148  virginica             0                 0                1  
149  virginica             0                 0                1  

[150 rows x 8 columns]
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2   
1                  4.9               3.0                1.4               0.2   
2                  4.7               3.2                1.3               0.2   
3                  4.6               3.1                1.5               0.2   
4                  5.0               3.6                1.4               0.2   
..                 ...               ...                ...               ...   
145                6.7               3.0                5.2               2.3   
146                6.3               2.5                5.0               1.9   
147                6.5               3.0                5.2               2.0   
148                6.2               3.4                5.4               2.3   
149                5.9               3.0                5.1               1.8   

     label_setosa  label_versicolor  label_virginica  
0               1                 0                0  
1               1                 0                0  
2               1                 0                0  
3               1                 0                0  
4               1                 0                0  
..            ...               ...              ...  
145             0                 0                1  
146             0                 0                1  
147             0                 0                1  
148             0                 0                1  
149             0                 0                1  

[150 rows x 7 columns]
```

```
# 8 - 4
X = df[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']]
X = np.asarray(X)
print(df)

y = df[['label_setosa', 'label_versicolor', 'label_virginica']]
y = np.asarray(y)
print(df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

     label_setosa  label_versicolor  label_virginica
0               1                 0                0
1               1                 0                0
2               1                 0                0
3               1                 0                0
4               1                 0                0
..            ...               ...              ...
145             0                 0                1
146             0                 0                1
147             0                 0                1
148             0                 0                1
149             0                 0                1

[150 rows x 7 columns]
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

     label_setosa  label_versicolor  label_virginica
0               1                 0                0
1               1                 0                0
2               1                 0                0
3               1                 0                0
4               1                 0                0
..            ...               ...              ...
145             0                 0                1
146             0                 0                1
147             0                 0                1
148             0                 0                1
149             0                 0                1

[150 rows x 7 columns]
```

```
# 8 - 14
import tensorflow_datasets as tfds
import tensorflow as tf
```

```
# 8 - 15
(train_data, test_data), info = tfds.load(
    'imdb_reviews/subwords8k',
    split=(tfds.Split.TRAIN, tfds.Split.TEST),
    with_info=True,
    as_supervised=True
)

padded_shapes = ([None], ())
train_batches = train_data.shuffle(1000).padded_batch(10, padded_shapes=padded_shapes)
test_batches = test_data.shuffle(1000).padded_batch(10, padded_shapes=padded_shapes)

for _ in range(3):
  train_batch, train_labels = next(iter(train_batches))
  print(train_batch.numpy())
  print('₩n')
```

```
WARNING:absl:TFDS datasets with text encoding are deprecated and will
[[ 636    2   12 ...    0    0    0]
 [1179    6  208 ...    0    0    0]
 [ 387  381   57 ...    0    0    0]
 ...
 [7963 3923   34 ...    0    0    0]
 [4728 7974   14 ...    0    0    0]
 [ 133 2788    2 ...    0    0    0]]


[[2589 3171   36 ...    0    0    0]
 [  69    9  278 ... 7994 7972 7975]
 [  62    9    4 ...    0    0    0]
 ...
 [2988 1416   21 ...    0    0    0]
 [ 636    2   12 ...    0    0    0]
 [ 135 7968    8 ...    0    0    0]]


[[ 768  284 8002 ...    0    0    0]
 [  12  109 7968 ...    0    0    0]
 [  12  580   14 ...    0    0    0]
 ...
 [ 133  296   27 ...    0    0    0]
 [7924   14   32 ... 5695  136 7975]
 [  12  321    1 ...    0    0    0]]
```

```
dataset = tf.data.Dataset.range(11)
for window in dataset:
  print(window)
print('₩n')

dataset = dataset.map(lambda x: tf.fill([tf.cast(x, tf.int32)], x))
for window in dataset:
  print(window)
print('₩n')

dataset = dataset.padded_batch(2, padded_shapes=(None,))
for window in dataset:
  print(window)
print('₩n')
```

```
tf.Tensor(0, shape=(), dtype=int64)
tf.Tensor(1, shape=(), dtype=int64)
tf.Tensor(2, shape=(), dtype=int64)
tf.Tensor(3, shape=(), dtype=int64)
tf.Tensor(4, shape=(), dtype=int64)
tf.Tensor(5, shape=(), dtype=int64)
tf.Tensor(6, shape=(), dtype=int64)
tf.Tensor(7, shape=(), dtype=int64)
tf.Tensor(8, shape=(), dtype=int64)
tf.Tensor(9, shape=(), dtype=int64)
tf.Tensor(10, shape=(), dtype=int64)


tf.Tensor([], shape=(0,), dtype=int64)
tf.Tensor([1], shape=(1,), dtype=int64)
tf.Tensor([2 2], shape=(2,), dtype=int64)
tf.Tensor([3 3 3], shape=(3,), dtype=int64)
tf.Tensor([4 4 4 4], shape=(4,), dtype=int64)
tf.Tensor([5 5 5 5 5], shape=(5,), dtype=int64)
tf.Tensor([6 6 6 6 6 6], shape=(6,), dtype=int64)
tf.Tensor([7 7 7 7 7 7 7], shape=(7,), dtype=int64)
tf.Tensor([8 8 8 8 8 8 8 8], shape=(8,), dtype=int64)
tf.Tensor([9 9 9 9 9 9 9 9 9], shape=(9,), dtype=int64)
tf.Tensor([10 10 10 10 10 10 10 10 10 10], shape=(10,), dtype=int64)

tf.Tensor(
[[0]
 [1]], shape=(2, 1), dtype=int64)
tf.Tensor(
[[2 2 0]
 [3 3 3]], shape=(2, 3), dtype=int64)
tf.Tensor(
[[4 4 4 4 0]
 [5 5 5 5 5]], shape=(2, 5), dtype=int64)
tf.Tensor(
[[6 6 6 6 6 6 0]
 [7 7 7 7 7 7 7]], shape=(2, 7), dtype=int64)
tf.Tensor(
[[8 8 8 8 8 8 8 8 0]
 [9 9 9 9 9 9 9 9 9]], shape=(2, 9), dtype=int64)
tf.Tensor([[10 10 10 10 10 10 10 10 10 10]], shape=(1, 10), dtype=int64)
```